"Graduated Embodiment for Sophisticated Agent Evolution and Optimization"

Position Paper on Scalability of Evolutionary Computation

Arthurine Breckenridge, Mark Boslough, and Michael Peters

Sandia National Laboratories
PO Box 5800, MS 1004, Albuquerque, NM 87185
[arbreck, mbboslo, mpeters] @sandia.gov

GECCO Workshop
June 26, 2004 – Seattle Washington, USA

Graduated Embodiment for Sophisticated Agent Evolution and Optimization

Goal: *To develop methods for graduated evolution for embodied agent optimization.*

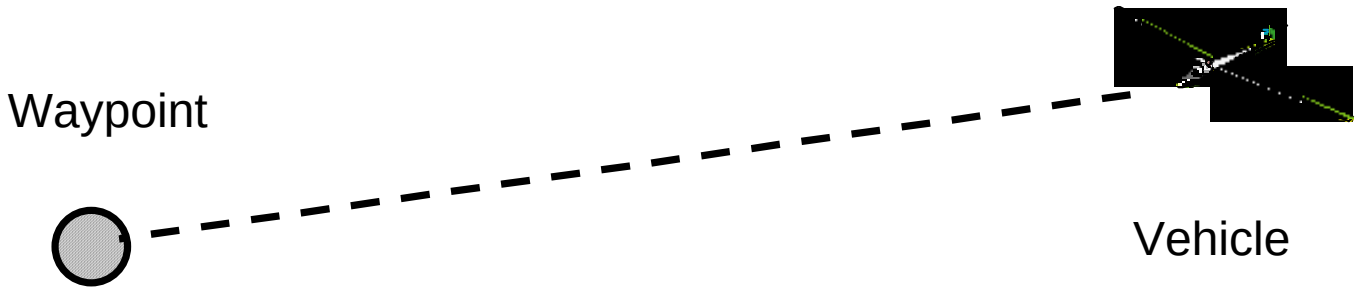
Method: *Evolve agent behaviors in a hierarchical fashion by staging fitness through multiple levels of fidelity.*

Project Accomplishments

- *Implemented Staged Optimization*
  - *Showed that staged methods exhibit improved scaling (as expected).*
- *Implemented Pruning*
  - *Pruned trees demonstrated reduced computational load (as expected).*
- *Combined hand-coded and machine-generated behaviors*
  - *Can build algorithms that are truly human-machine collaborations.*
- *Converted GP code to C++*
  - *Allows for improved modularization; shifts burden off user.*
- *Incorporated GP behaviors into UMBRA*
  - *High fidelity environment for high fidelity behaviors.*

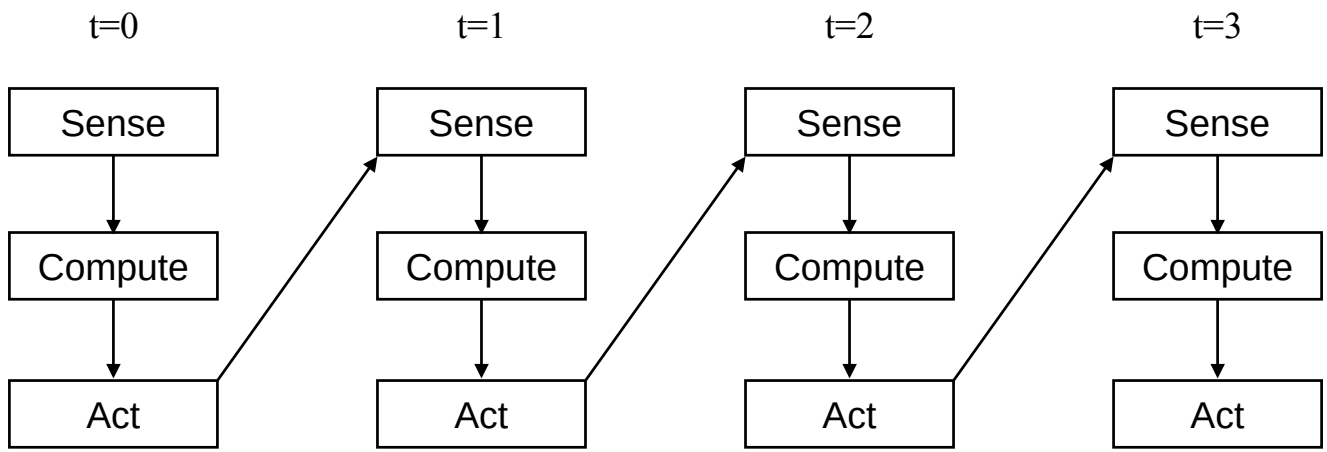
Adaptive Waypoints

This encapsulation is intended to bridge the gap between toy problems and real problems



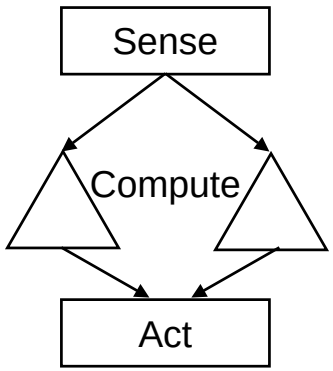
Our method to develop levels of integration:
Vertical, Horizontal, Co-Evolution, Collective Behavior

General implementation concept: Use GP to design “compute” step

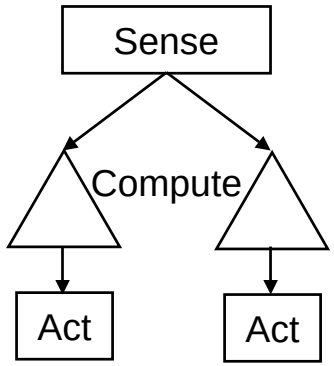


Classification of Building Blocks

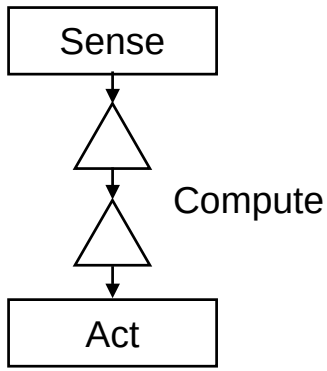
Horizontal Sequential



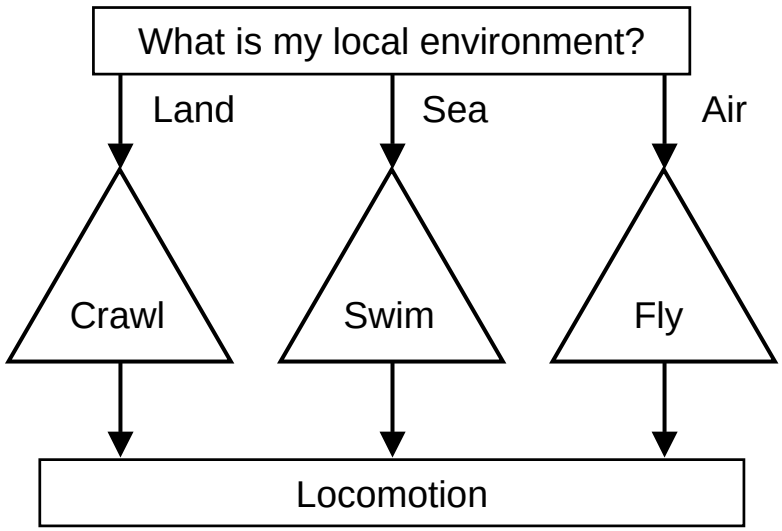
Horizontal Parallel



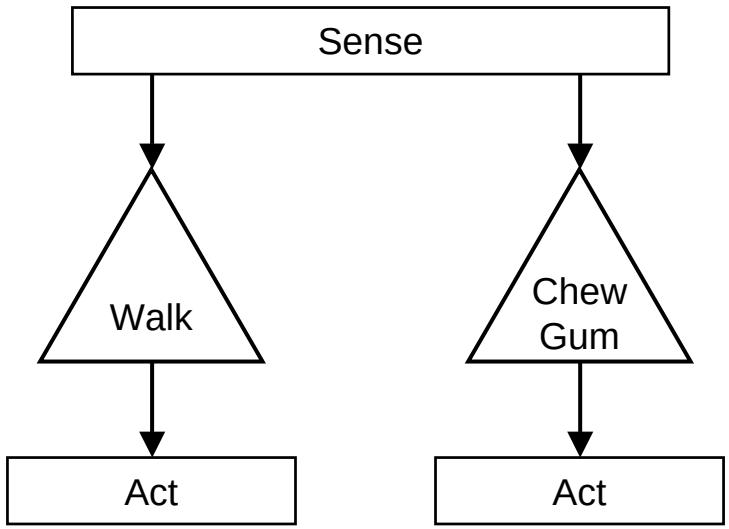
Vertical



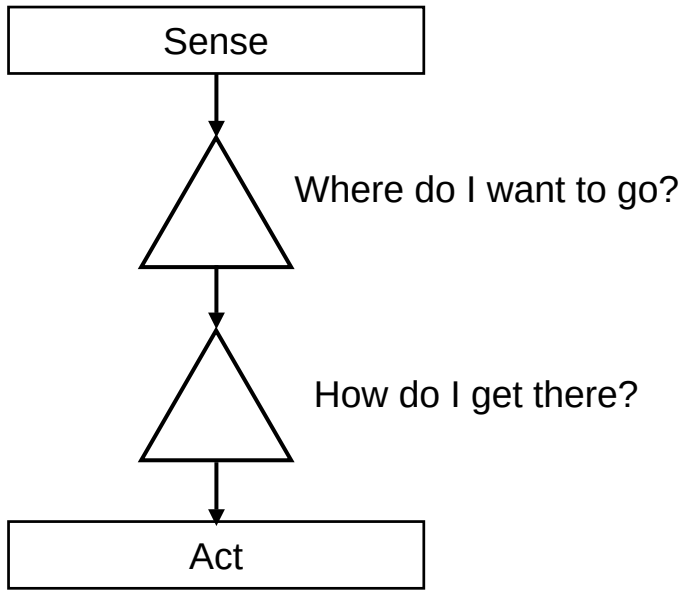
Example: Horizontal Sequential



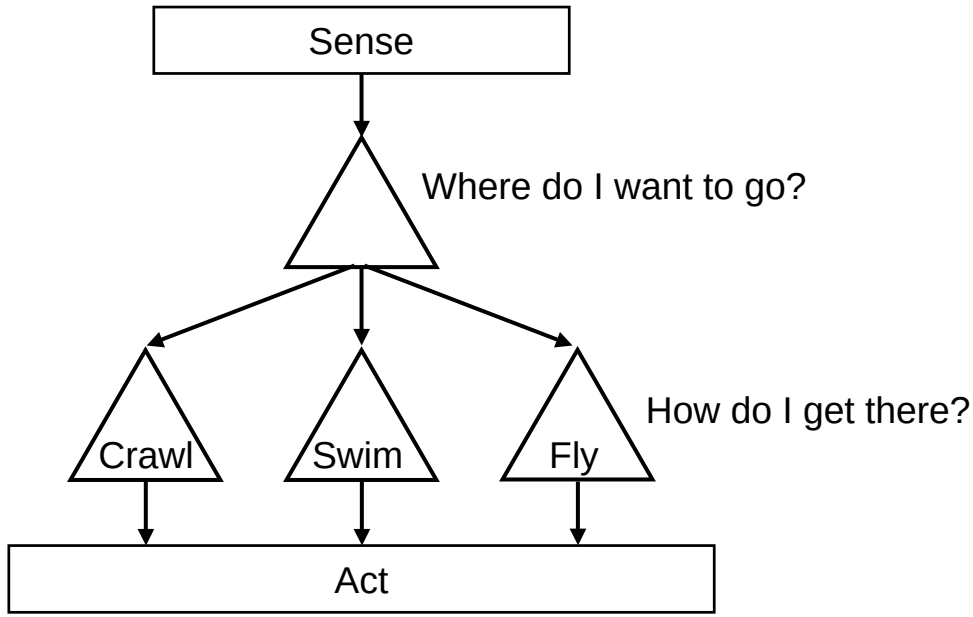
Example: Horizontal Parallel



Example: Vertical



Example: Combined



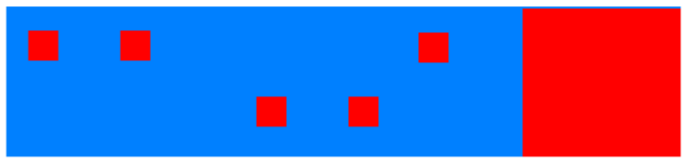
Benefits of Converting code from C to C++: Modularization

C++ reduces and modularizes changes in code

Change optimization in C code

Code that stays the same

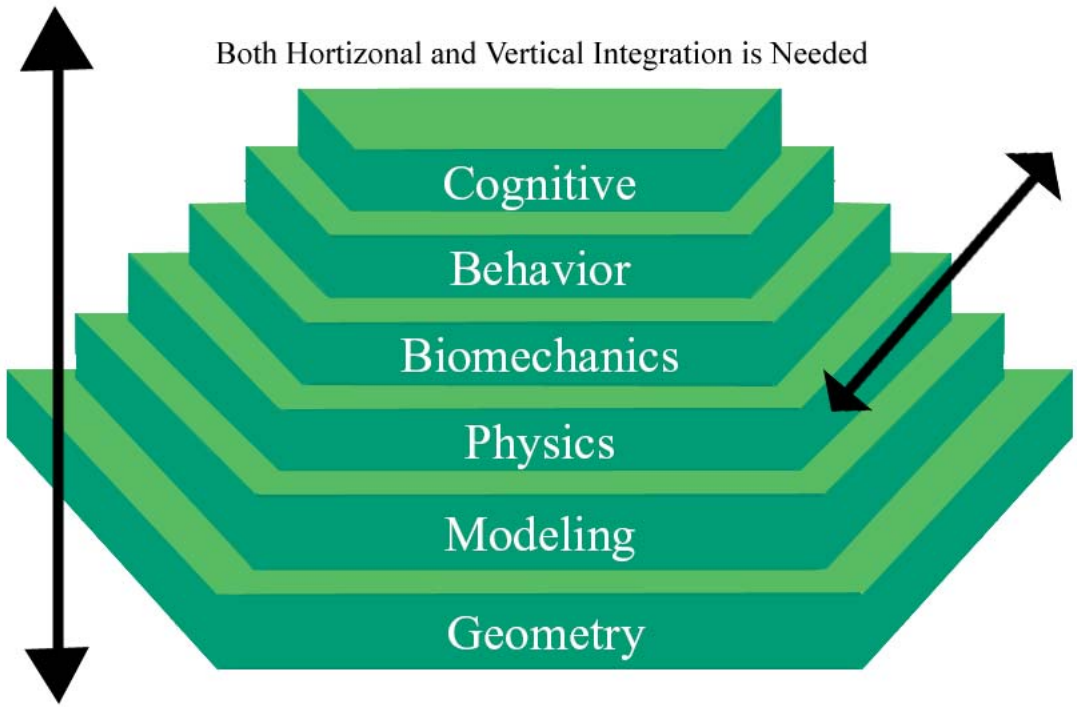
Code that changes



More changes in more places mean more bugs and longer development time

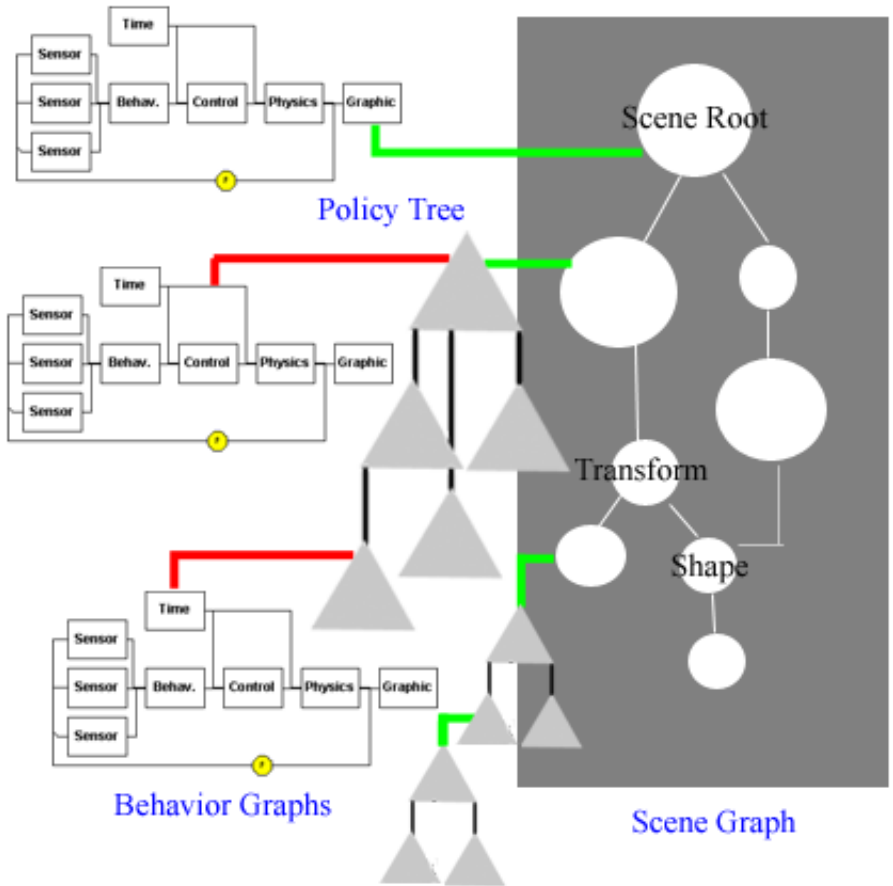
Change optimization in C++ code

Multiple levels are needed: both horizontal and vertical

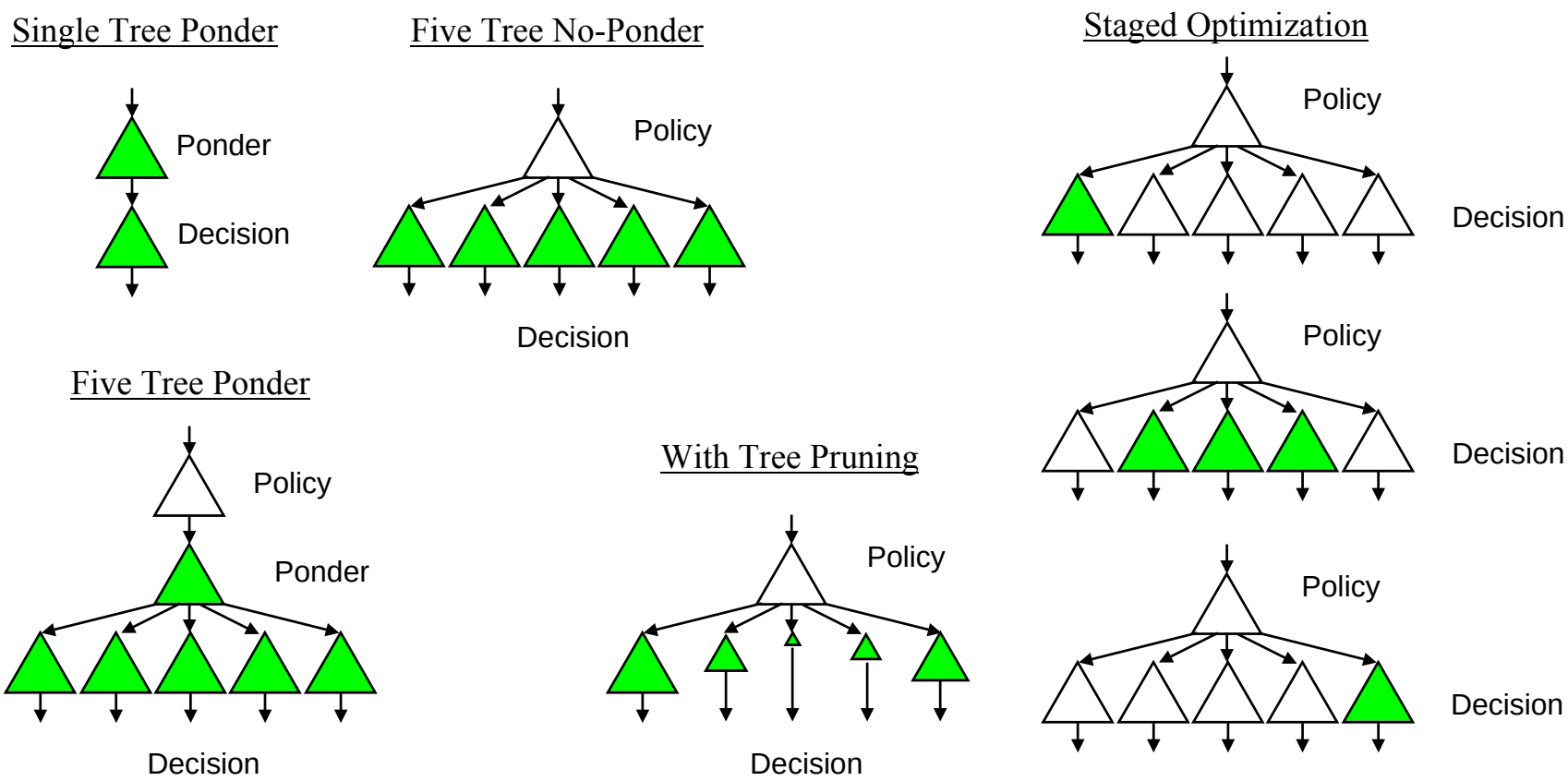


*Modeling Building Blocks according to Funge, Terzopoulos, 1999

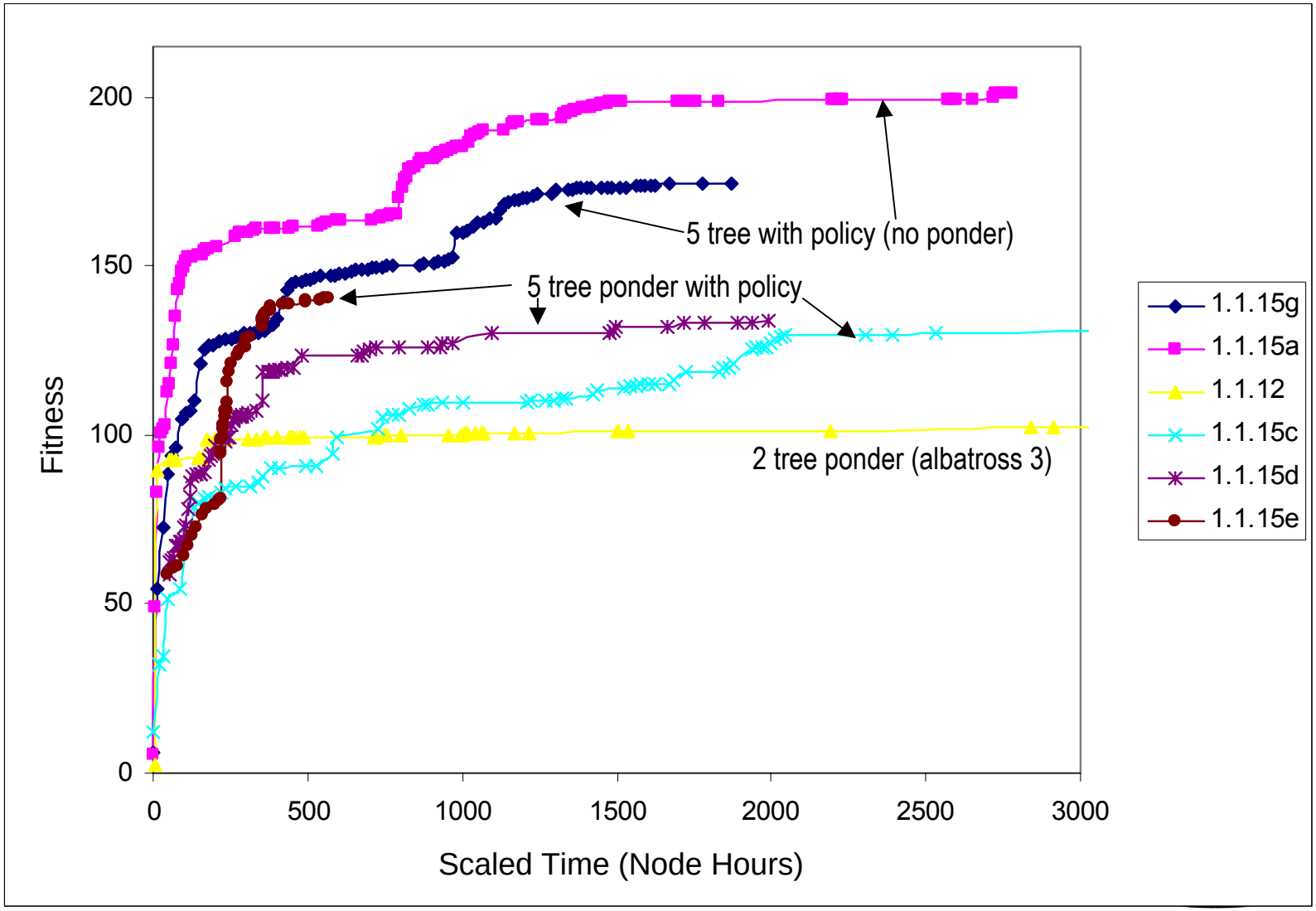
We are developing tools to visualize evolved behaviors



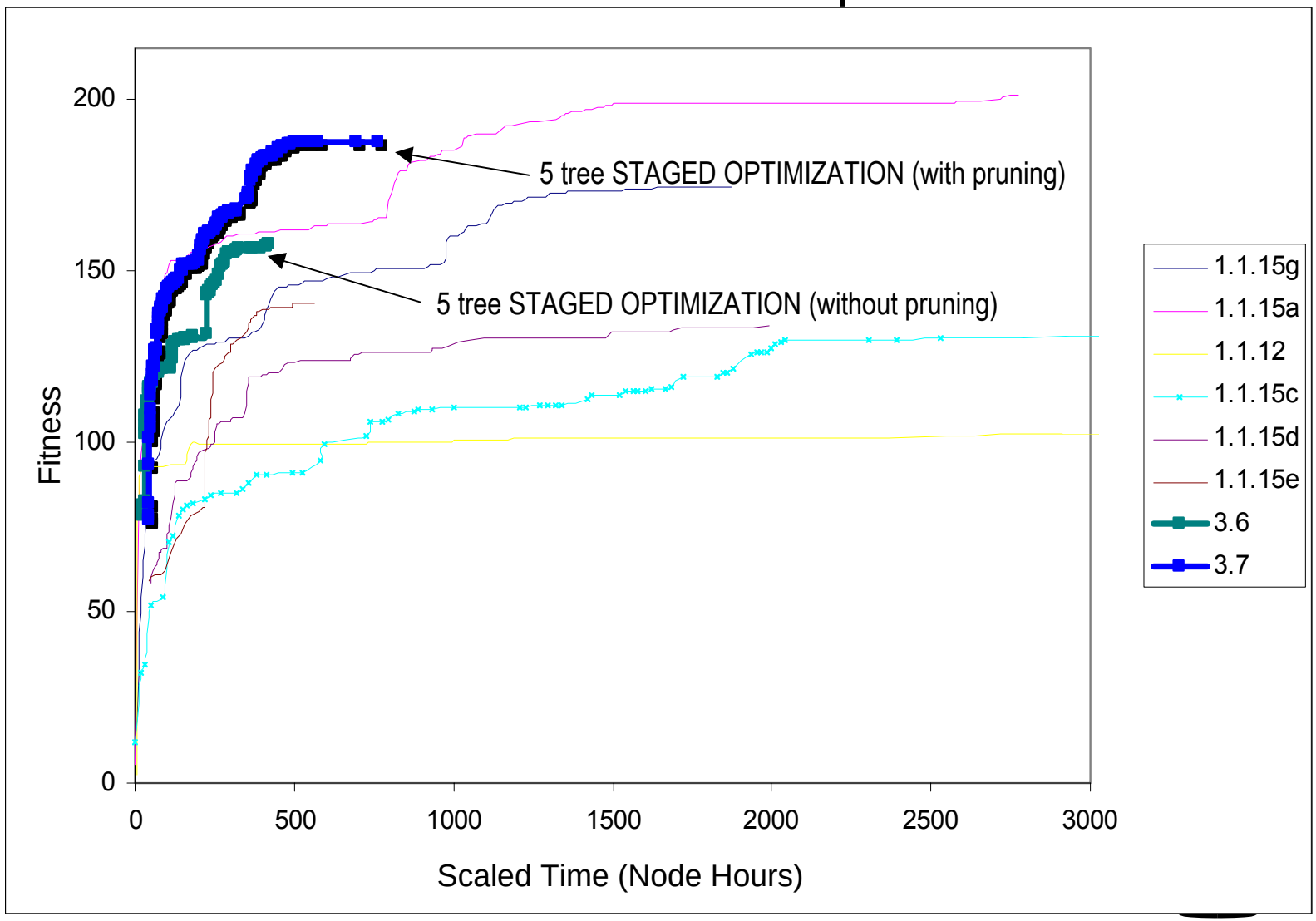
Benchmark problem optimization approaches



Results for benchmark problem



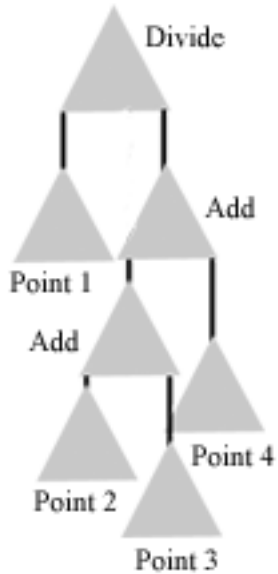
Results for benchmark problem



Real-world example: Image data collection

Sense Compute Act (Machines)

Pixel Neighborhood

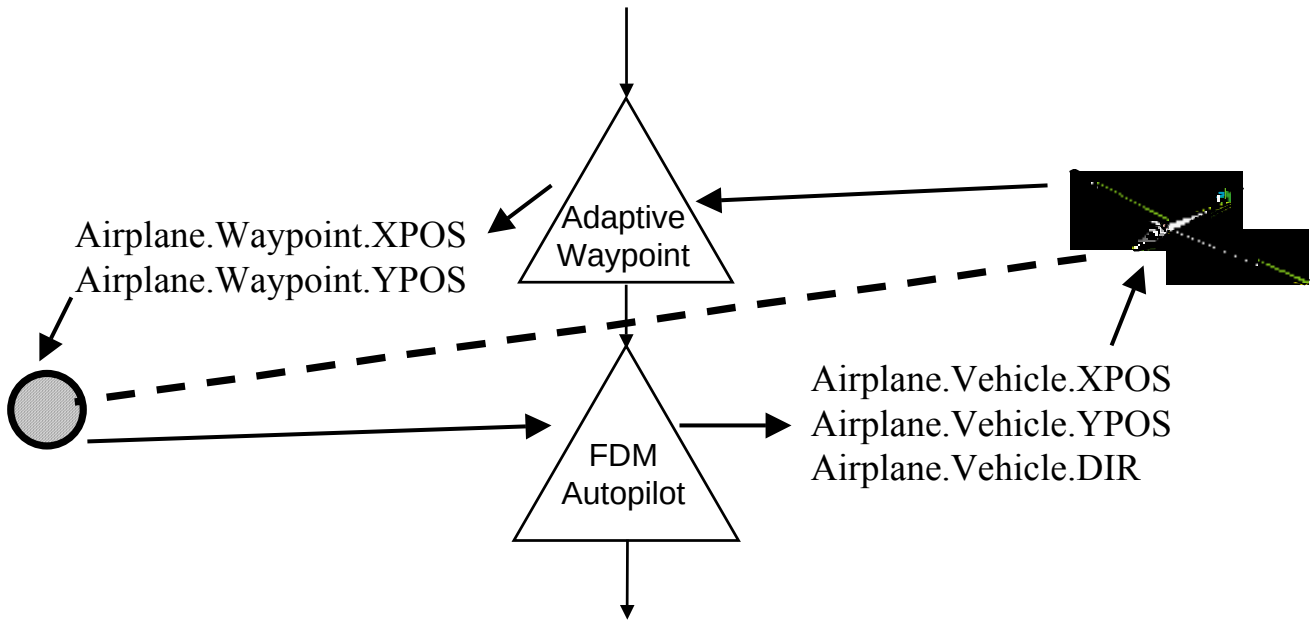


If Value < Target Value

MOVE (C)

Pereceptual Cognitive Behavorial (Humans)

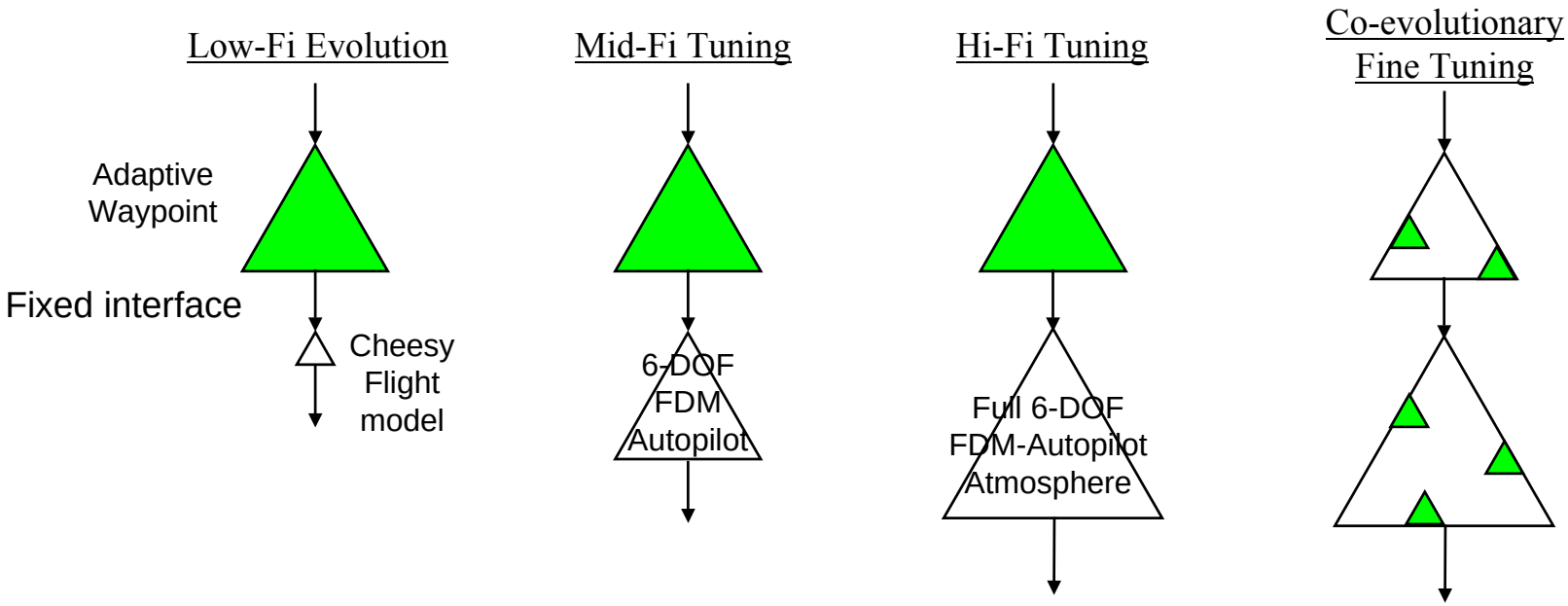
Graduated Embodiment example: adaptive waypoint



Allows Vertical, Horizontal, Co-Evolutionary, Collective Behavior

Graduated Embodiment example: adaptive waypoint

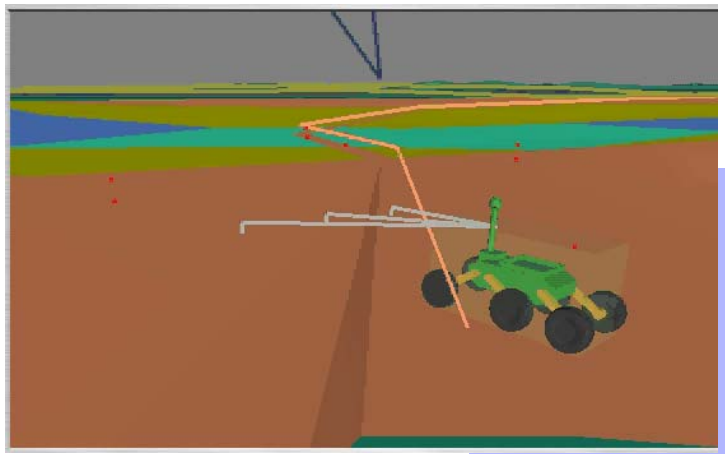
We have chosen this method to develop usable behaviors



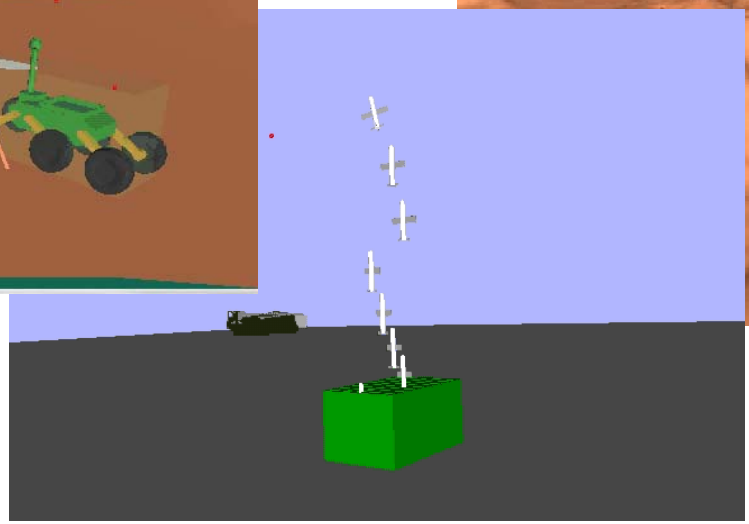
This is the essence of Graduated Embodiment

Next: Apply GP to Umbra's hard engineering

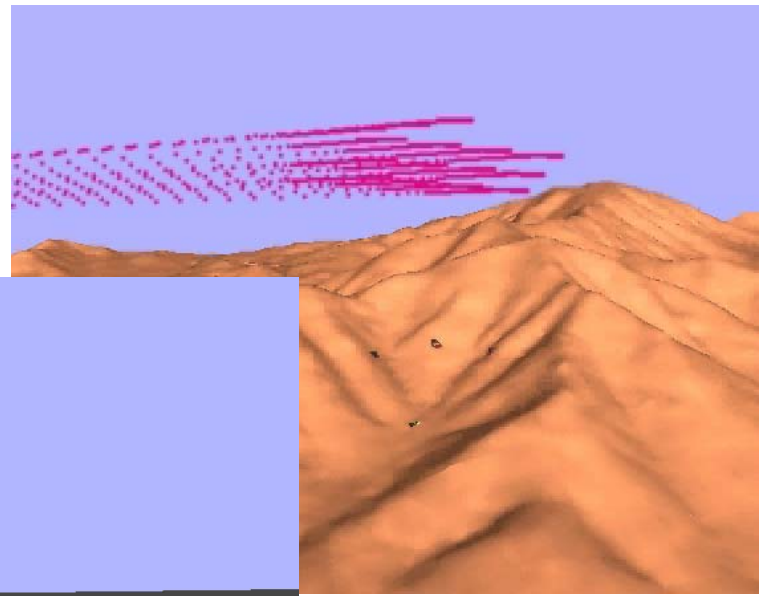
Can the adaptive waypoint concept be generalized to n-demensional parameter space??



Simple target on terrain



Netfires at launch

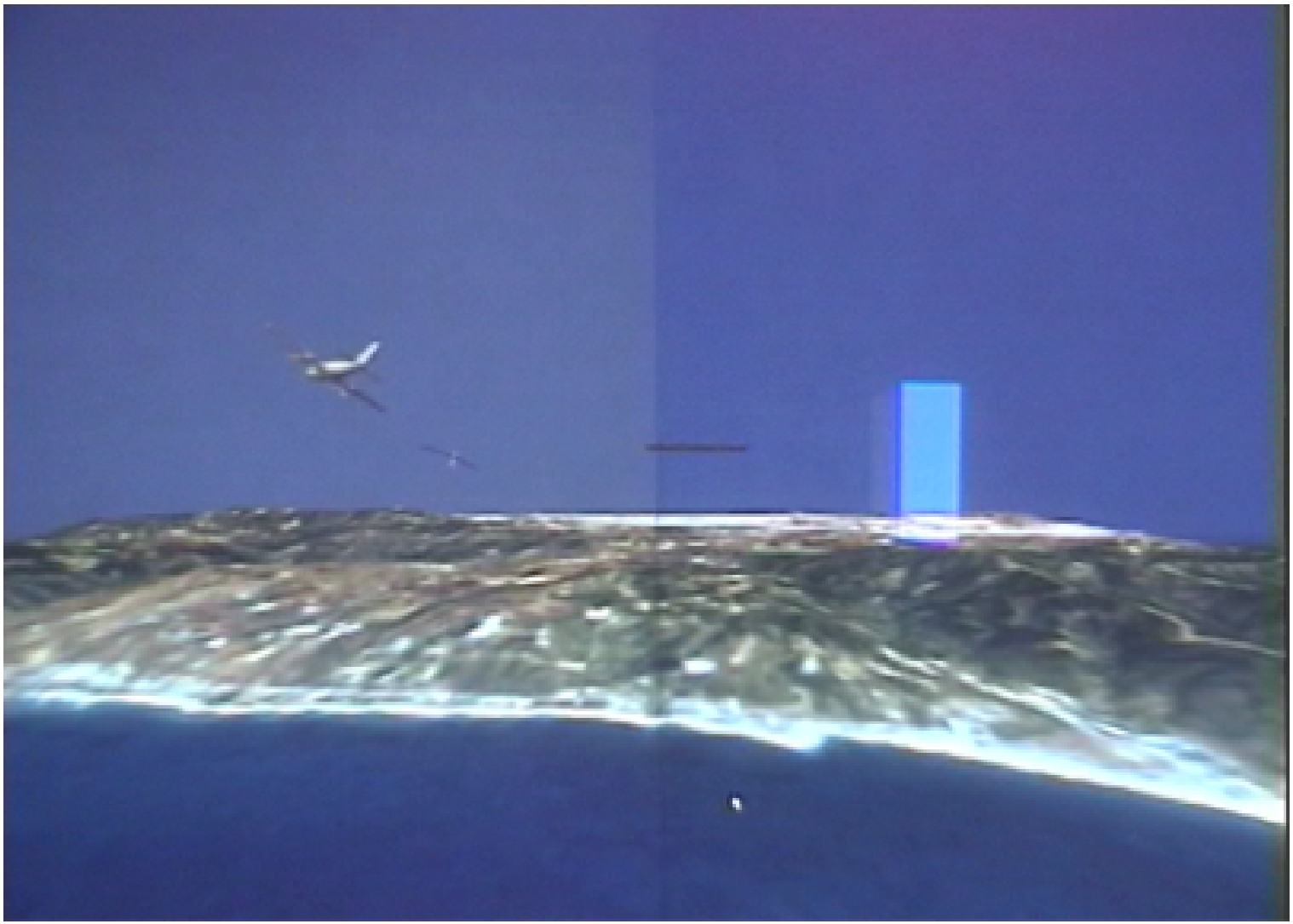


Netfires target seeking

Why are we developing evolutionary computing technology?

- *Well-suited for robotics applications.*
- *Simulation pushes Umbra "multiple-fidelity" capability.*
- *Provides context for intelligent machine systems.*
- *Likely to provide insight into cognition processes.*
- *Uses* behavioral biomimetics *to derive insight from nature.*

Umbra Application of Genetic Programming
Running on 2X2 Tile in VIEWS Corridor Building 880/A1



2056 X 2048 Resolution taken with 720X 480 Camera